Journal of Theoretical and Applied Information Technology
15th January 2021. Vol.99. No 1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SECURITY REQUIREMENTS TEMPLATE-BASED APPROACH TO IMPROVE THE WRITING OF COMPLETE SECURITY REQUIREMENTS

[1] NURIDAWATI MUSTAFA, [2] MASSILA KAMALRUDIN, [3] SAFIAH SIDEK

[1]Senior Lecturer, Department of Software Engineering, Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, Malaysia

[2]Professor, Innovative Software System and Service Group (IS3), Universiti Teknikal Malaysia Melaka, Malaysia

[3]Associate Professor, Innovative Software System and Service Group (IS3), Universiti Teknikal Malaysia Melaka, Malaysia

E-mail: [1]nuridawati@utem.edu.my, [2]massila@utem.edu.my, [3]safiahsidek@utem.edu.my

ABSTRACT

Writing quality security requirements contributes to the success of secure software development. It has been a common practice to include security requirements in a software system after the system is defined. Thus, incorporating security requirements at a later stage of software development will increase the risks of security vulnerabilities in software development. However, the process of writing security requirements is tedious and complex. Although significant work can be found in the field of requirements elicitation, less attention has been given for writing complete security requirements. It is still a challenge and tedious process for requirements engineers (REs) to elicit and write complete security requirements that are derived from natural language. This is due to their tendency to misunderstand the real needs and the security terms used by inexperienced REs leading to incomplete security requirements. Motivated from these problems, we have developed a prototype tool, called SecureMEReq to improve the writing of complete security requirements. This tool provides four important key-features, which are (1) extraction of security requirements components from client-stakeholders; (2) validation of security requirements probability density and security requirements syntax density; (3) checking the security requirements and key-structure components; and (4) validation of completeness prioritization. To do this, we used our pattern libraries: SecLib and SRCLib to support the automation process of elicitation, especially in writing the security requirements. To evaluate our approach and tool, we have conducted completeness tests to compare the completeness of writing security requirements through the results provided by SecureMEReq and manual writing. Our evaluation results show that our prototype tool is capable to facilitate the writing of complete security requirements and useful in assisting the REs to elicit the security requirements.

Keywords: *Tool Security Requirements, Template-Based Approach, Security Requirements Completeness, Template-Based Density, Syntax Density*

1. INTRODUCTION

Capturing complete security requirements is crucial for the development of a secure software because incompletely defined and poor elicited security requirements may result in costly development failure [1]. Further, incomplete security requirements could lead to generating incorrect non-functional security requirements [2]. At present, when capturing security requirements from clients, Requirement Engineers (RE) often uses some forms of natural language, written either by clients or themselves. These requirements are captured from the discussion and negotiation between both parties; clients and the RE. However, due to the ambiguities and complexities of natural language [3][4] and the process of capturing, these requirements often have incompleteness. RE also faced problems in eliciting consistent security compliance requirements from the clients-

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

stakeholders as they misunderstood the real needs and the security terms used [5].

Security requirements can be defined as a system specification of its required security, such as the specification towards types and levels of protection, necessary for the data, information, and application of the systems. It is also categorized into functional and non-functional requirements [6]. Examples of security requirements are authentication requirements, authorization requirements, intrusion detection requirements, and many others [7].

A common approach for the inclusion of security within a software system is to identify the security requirements after a system is defined in the developed software. Thus, incorporating security requirements at the later stages of a software development increases the risks of introducing security vulnerabilities into the software [8]. This paper advocates [9], in which completeness checking needs to be done at the earliest stage of requirements engineering (RE) process, as shown in Figure 1. This approach would minimize the number of defects that would otherwise permeate further phases of software development [10].

This rest of the paper is organized as follows. In Section 2, the background and the motivations for this work are highlighted. Next, in Section 3, the overview of our template-based approach is described. In Section 4, we discuss the implementation of tool support. In Section 5, we explain our tool validation study methodology to evaluate the effectiveness of our approach. Then, we explain the threats of validity. The result and discussion of this paper is presented in Section 7. Finally, we conclude the paper with some remarks on our proposed approach and possible future work for improvement.

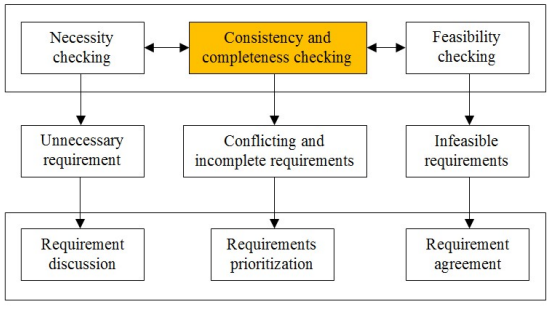


Figure 1: Requirement Analysis And Process Flow [9]

2. BACKGROUND AND MOTIVATIONS

Requirements elicitation is one of the complex processes as it involves many activities [11]. The activities in requirements elicitation involve a variety of techniques, methods, approaches, and tools for capturing complete requirements. There are a few works related to eliciting requirements, such as interview [10];[12], questionnaire [13];[14] and observation [15]. Security requirements elicitation is the most essential activity to gain understanding between the development team and the business team. It also maps the information to develop systems/applications in accordance to business and user needs, provided by stakeholders as high-level statements on features and functionalities. In order to address the security aspects, it is necessary for the business, development and security teams to understand the key sensitivities and business consequences caused by risk of security flaws. Developing a software usually follows the System Development Life Cycle (SDLC), which is a feed forward process; hence, any errors introduced in this phase will be spread to the next development phases. Thus, it is important to elicit security requirements at the very early stages [16].

There are several works found in writing security requirements (Refer to Table 1). However, there are a few gaps found in the existing works, which we categorized them as the method-related and people-related issues in security requirements elicitation.

Table 1: Comparison of security requirements elicitation techniques

Technique	SDLC		Validation	Security Requirements Template	Tool	Security Standards	Validation (M= Manual, SA=Semi-Auto A=Auto)
	Analysis	Design	(Y = Yes, N = No)				
Multilateral	√		N	N	N	N	-
UML-Based	√		N	N	N	N	-
UML-Based (Use-Case Driven)	√		Y	Y	Y	Y	SA
Goal-Oriented		√	Y	N	Y	Y	M
Problem Frame	√		N	N	Y	Y	-
Risk-Analysis	√		N	N	N	N	-
Common	√		Y	N	N	Y	M

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Technique	SDLC		Validation	Security Requirements Template	Tool	Security Standards	Validation
	Analysis	Design					(M= Manual, SA=Semi-Auto A=Auto)
	(Y = Yes, N = No)						
Criteria							
Essential Use Case	√		Y	N	Y	N	A
Resource Centric	√		N	N	N	N	-

The method-related issues include the lack of (1) checking on security requirements completeness; (2) security requirements templates; (3) security standards used as reference; and (4) automated tool for validation. Based on our study, one of the major challenges is the adaptation of proper methods for writing complete security requirements due to the lack of completeness checking and security template in previous research. Even though there are existing security requirements templates, none of the existing approaches are provided with density calculations and completeness status, which are important for determining requirements completeness level. Besides, there are still lacking of works that refer to security standards while developing their security requirements templates. In fact, the tedious writing process of security requirements requires high skill and experience requirements engineers to carry out the process. This is due to insufficient resource in terms of proper tool to support the writing security requirements process since most of the checking were conducted manually and without automated or semi-automated approaches for completeness checking. These problems lead to writing incomplete security requirements and disruptions of schedule and increment to project's expenditure [17] [18] [1].

While, the people-related issues consist of (1) inexperienced requirements engineers; (2) minimal involvement of technical team in defining security requirements; and (3) language barriers. We have found that most of requirement engineers lack of skill in terms of security related requirements due to the lack of training. This is supported by Salini and Kanmani (2012a), who argued that most of requirements engineers are poorly trained in security. In fact, those who have undergone the training were only given an overview of security architectural mechanisms, such as passwords and encryption rather than the actual security requirements [20].

The requirements engineers faced problems to elicit security requirements from the clients-stakeholders as there are instances of mismatch between the real needs and the security terms used (Houmb et. al., 2010; Banerjee et. al., 2015). Besides, the exclusion of technical team in the process causes the late identification of security-related requirements that may lead to writing incomplete security requirements. The security requirements are often inadequately understood and improperly specified, which is often due to the lack of security expertise and the lack of emphasis on security during the early stages of system development [21]. The language barrier between the requirement engineers and client-stakeholders is also identified as another obstacle. Ambiguities and complexities of natural language [3] [4] and the tedious process of capturing lead to misinterpretation and misunderstanding of the security requirements as well as the difficulty to reach mutual agreement among stakeholders, hence resulting in writing incomplete security requirements [22]. Based on the above issues, requirements engineers are still facing problems with writing incomplete security requirements although security requirements are important in building secure software.

In this section, we also summarize the tool comparison provided by security requirements elicitation techniques. Based on Table 2, there are only four tools from four techniques that provide a tool to support the writing of security requirements. In terms of tool validations, each tool performs specific aspects of validation. Specifically, the UML-Based (Use-case driven) validates consistency, the goal-oriented technique validates the correctness, consistency and completeness of the security, the common criteria-based technique validates the consistency and completeness only, while the essential use-case validates the correctness criteria only. All these validations are done manually, except the UML-Based technique. Out of these works, the goal-oriented works are based on NIST, particularly in Secure i* method. The UML-Based (use-case driven) is referred to ISO/IEC 27001:2013. Meanwhile, Problem Frame method refers to security standards, which is the ISO/IEC 15408 in security problem frames [23]. The common criteria-based by Mellado et al. (2007) also integrates the ISO/CC in their work.

Overall, most of tools lack of completeness checking functionalities. Yet, all of these tools do not provide the template for writing security requirements that can assist requirement engineers

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

in writing complete security requirements. Most of the tools discussed above do not use the security standard as their reference; hence, it can be inferred that so far, there has been limited tools for eliciting requirements with direct reference to security standards. It was found that there is only one tool that can automate the correctness checking during the security requirements elicitation process. Hence, it can be concluded that tools that support automated completeness checking is still lacking. Based on a comparison study between the manual and template-based approach, it was found that none of the tools have similar functionalities as the SecureMEReq. We also found that SecureMEReq is an automated tool that performs completeness validation and provides security requirements writing template based on three security standards, namely the ISO/IEC, NIST and Common Criteria.

Table 2: Security Requirements Elicitation Techniques and Tool Comparisons

Technique	Tool	Tool Type			Tool Validation			Template	Security Standards
		M	SA	A	C_R	C_N	C_M		
Multilateral	X				X	X	X	X	X
UML-Based	X				X	X	X	X	X
UML-Based (Use-Case Driven)	√		√		X	√	X	√	√
Goal-Oriented	√	√			√	√	√	X	√
Problem Frame	√				X	X	X	X	√
Risk-Analysis	X				X	X	X	X	X
Common Criteria	X	√			X	√	√	X	√
Essential Use Case	√			√	√	X	X	X	X
Resource Centric	X				X	X	X	X	X
Template-Based (SecureMEReq)	√			√			√	√	√

C_R - Correctness, C_N - Consistency, C_M – Completeness, M- Manual, SA-Semi-Auto, A-Auto

3. TEMPLATE-BASED APPROACH

In relation to the gaps highlighted in Section 2, this research aimed to propose a security requirements template-based approach to improve the clarity of requirements that can lead to writing complete security requirements. This work is based on the research question below:

"How does the template-based approach help in writing complete security requirements?"

We proposed the overall template-based approach as illustrated in Figure 2.

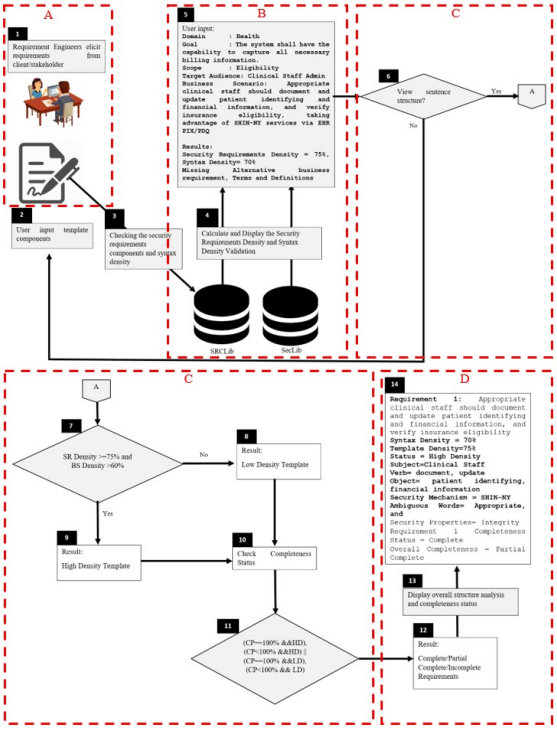


Figure 2: An Overview of Security Requirement Template-Based Approach

The approach comprises four main steps. Step A comprises of three processes (Process 1-3). In this step, the security requirements components are extracted, in which the RE elicits textual natural language requirement from the clients/stakeholders during requirements gathering. Then, the RE enters the security requirements components in the tool editor. Two aspects will be analysed from the textual security requirements. Firstly, each requirement is analysed using security requirements components from SRCLib. Secondly, the business scenario is analysed based on syntax analysis from SecLib library. Additionally, the purpose of the searching process from the SecLib library is to find the associated security keywords and mechanism using keyword matching. The function of these two aspects is to confirm that the security requirements follow the template and sentence structure of security requirements.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Step B performs the validation of security requirements probability density and security requirements syntax density in process 4 and 5. Here, the tool calculates the density for security requirements components and business scenario based on our security requirements pattern library SRCLib and SecLib. For further checking, RE can also view the analysis for each security requirement sentence structure.

Step C focuses on checking the security requirements and key-structure components, and it consists of five steps, which starts from step 6 until 11. Here, the tool displays the status of the security requirements density, whether it has high or low density. The subject, verb, object, security mechanism, ambiguity words and security properties will be displayed. Besides, the completeness status for each requirement is displayed. Additionally, the missing components will be highlighted. RE will have the ability to edit/update the input and choose the option whether to edit/update the original input.

Finally, in D, we validate the completeness prioritization, where the tool displays the overall result of the security requirements completeness level, either it is "Complete", "Partial Complete" or "Incomplete". The results will help RE to give early status on the level of completeness for their written security requirements.

To implement our approach in Figure 2, we developed two pattern libraries: (1) the Security Requirements Library (SecLib) for security requirements taxonomy and key textual tree structure, and (2) the Security Requirements Completeness Library (SRCLib) for security requirements probability density, syntax density and completeness prioritization.

To do that, we have conducted two main processes to design the template-based approach. We conducted analysis of requirements and semi-structured interview with requirement engineering experts from the industry, as illustrated in Figure 3. The aim of these processes is to discover the security requirements writing process practices and to reveal any problems or issues arisen during the writing the security requirements process in industry.

We categorized the security requirements knowledge into three parts: i) Basic 2) Applied 3) Advanced. The basic knowledge is defined as a form of foundation of security requirements such as security requirements properties. Whereas, the applied is the knowledge we acquire from the analysis of requirements and security standards. Meanwhile, the advanced specific knowledge we acquire from the semi-structured interview with RE practitioners.

Referring to Figure 3, firstly, we collected business requirements and security standards. Then, we conducted analysis of requirements activity to elicit basic and applied security requirements knowledge, for instances, knowledge on security requirements properties, security requirements taxonomy, security requirements syntax tree structure, security requirements syntax library, security requirements writing template and security requirements density calculation.

Next, to acquire security requirements advanced knowledge, we need to capture knowledge from the experts who are involved in requirements engineering field. Therefore, we conducted semi-structured interview. From the semi-structured interview, we obtained the security requirements advanced knowledge, which are the security requirements template components, their advanced practices in elicitation, and early evaluation on our automated tool. Then, we incorporated basic, applied and advanced knowledge to structure the generic security requirements template-based approach.

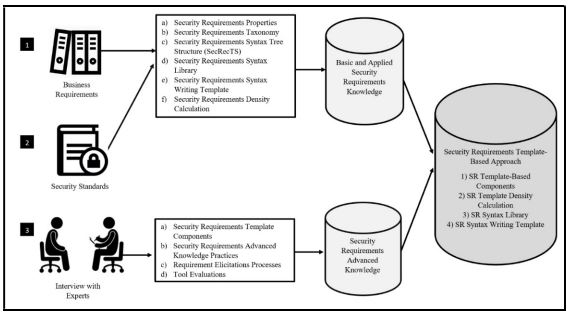


Figure 3: Processes of Designing Template-Based Approach

4. IMPLEMENTATION OF TOOL SUPPORT

We have developed a prototype tool, called SecureMEReq using PHP programming language and adopted Model-View-Controller (MVC) design pattern and three-tier architecture. MVC design pattern was implemented to develop a platform-independence software application that supports different platforms, such as mobile devices, tablets, and different browsers on different operating system. As shown in Figure 4, MVC pattern divides

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

an interactive application into three components: *Model*, *View* and *Controller*.

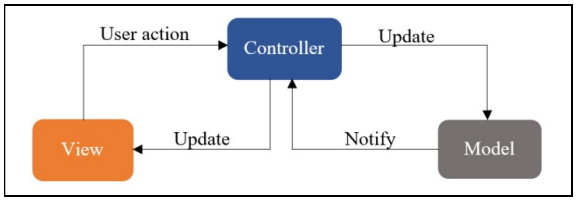


Figure 4: The MVC Design Pattern

The *Model* manages data and business logic of the application, the *View* is responsible for presentation of data or model and displays it to the user through browser, and the *Controller* manages the communication between the model and view. The development of SecureMEReq was adapted from the works of [25], [26] and the identification of the associated security elements was based on the definitions from the basic security services.

Figure 5 illustrates the high-level architecture of the SecureMEReq tool that comprises three tiers; presentation, business processing, and data management layer. The layout is three-tier architecture, where each layer is separated from each other. This independency allows for better performance, easier maintenance and more scalable architecture [27].

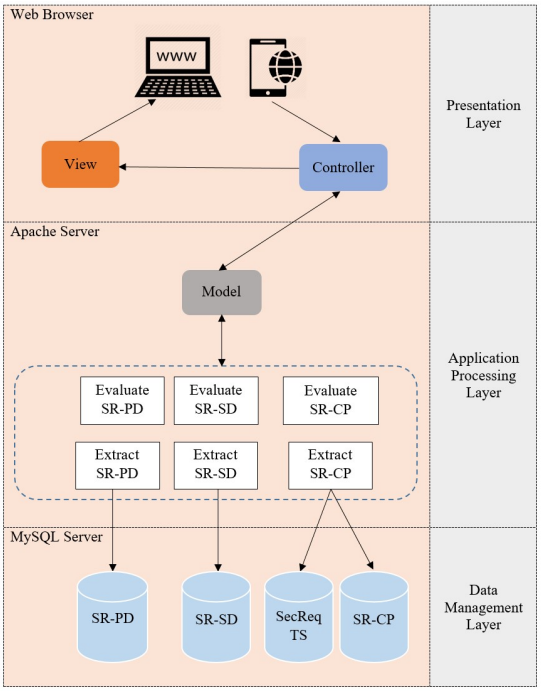


Figure 5: SecureMEReq High Level Architecture

The presentation layer handles the interaction between the users and the system. The *View* and *Controller* exist in the presentation layer. Here, a web client from any platform such as an iPad, mobile phone or desktop can request to access the SecureMEReq tool. The user interacts with the SecureMEReq tool through the *Controller* component. The *Controller* that contains the client-side scripting, handles the http request processing and business logic of the tool. It receives user input as events and translates them into service request for the *Model* or the *View*. When a user accesses the SecureMEReq, the scripts in the *Controller* will determine the type of browser and device used by the user. Then, it will request the correct view from the *View* component. Each view has associated controller component. Next, the *View* component will make requests from the *Model* to fetch the data from business and data layer and display the information to the user.

At the business processing layer, the Apache server hosted the PHP implementation for the main event handlers of SecureMEReq. This contains the key elements for the extraction of security requirements components from the textual requirements, extraction of business scenario components at the SecureMEReq's template editor, analysis and evaluation of the template-based component and business scenario syntax from pattern library and completeness prioritization analysis.

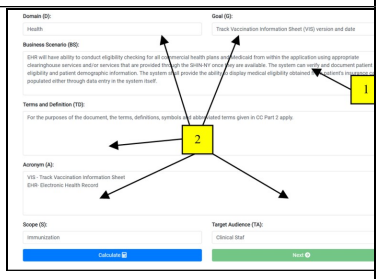
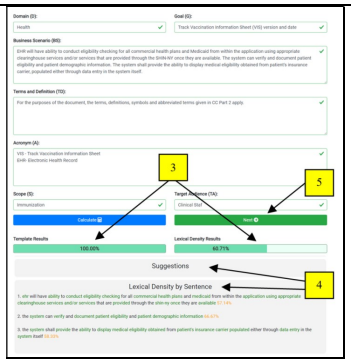
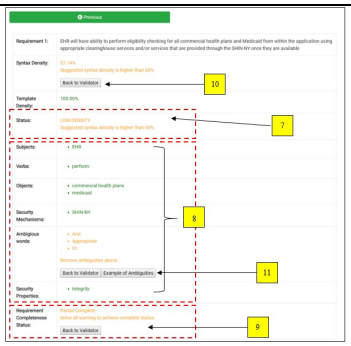
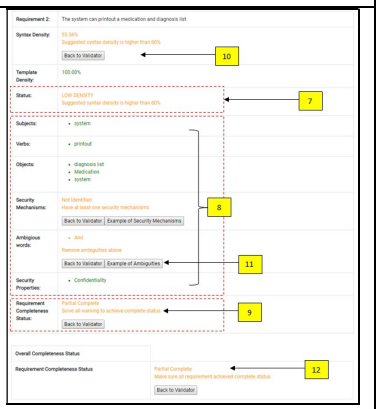
At the data management layer, MySQL database server contains the security requirements libraries and density library.

As overall, our tool provides the (1) extraction of security requirements components from client-stakeholders; (2) validation of security requirements probability density and security requirements syntax density; (3) checking the security requirements and key-structure components; and (4) validation of completeness prioritization. Table 3 below shows the mapping between proposed template and tool developed.

Journal of Theoretical and Applied Information Technology
15th January 2021. Vol.99. No 1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 3: Tool and Template-based Mapping

No.	Template	Tool Implementation
A.	Extraction of security requirements components	
B.	Validation of Security requirements probability density and security requirements syntax density	
C.	Checking the security requirements and key-structure components	
D.	Validation of completeness prioritization	

We demonstrated the features of our tool using the user persona as per described below:

John, a requirements engineer would like to validate the requirements provided by the client-stakeholder using SecureMEReq. He sits with Lewis, who is the project manager to validate the requirements, which he had captured earlier. As shown in Figure 6, firstly, he starts with extraction of security requirements components. He inserts the requirements in the form of business scenario in the text editor (1). Besides, he also needs to insert several security requirements components, which are the domain, goal, terms and definitions, acronym, scope and target audience as in (2).

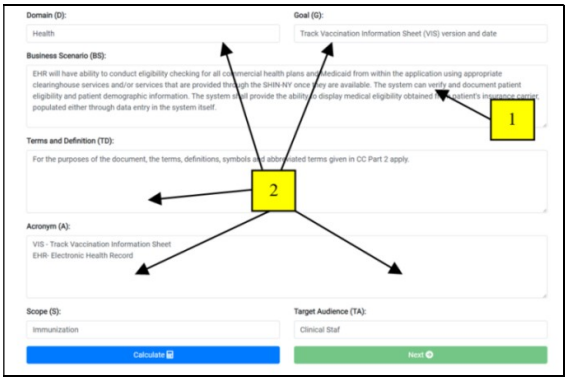


Figure 6: Extraction of Security Requirements Components

From there, as shown in Figure 7, he clicks the "Calculate" button to generate the density for security requirements components and syntax density (3). Here, John and Lewis will validate the security requirements probability density and security requirements syntax density. Then, John can view the security requirements probability density and syntax density results (4). If John is unhappy with the result, he can edit/update the inputs and recalculate, if needed. Besides, John and Lewis can review the "Suggestion" and "Lexical Density by Sentence" (5). In order to allow Lewis to get better understanding of the requirements structure, he then clicks the "Next" button to review the analysis of security requirements (6).

Journal of Theoretical and Applied Information Technology
15th January 2021. Vol.99. No 1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

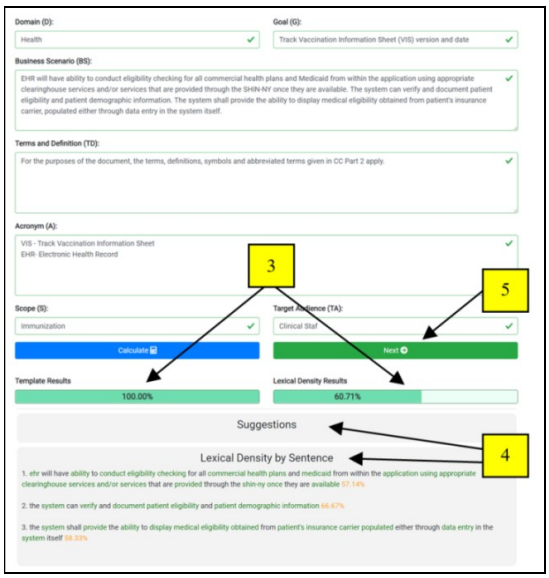


Figure 7: Validation of Security requirements probability density and security requirements syntax density

In Figure 8, John and Lewis can check the security requirements and key-structure components. Here, John and Lewis can validate each requirement density status and structure, such as the Subject, Object, Verb, Security Mechanism, Ambiguous Words used, Security Properties and the completeness status for each requirement (7)(8). They can also view the examples of each component if needed (11). He can view the completeness for each requirement (9) and Lewis can decide whether to proceed with the requirements or amend it (10). Finally, in Figure 9, they can validate the completeness prioritization for overall completeness (12).

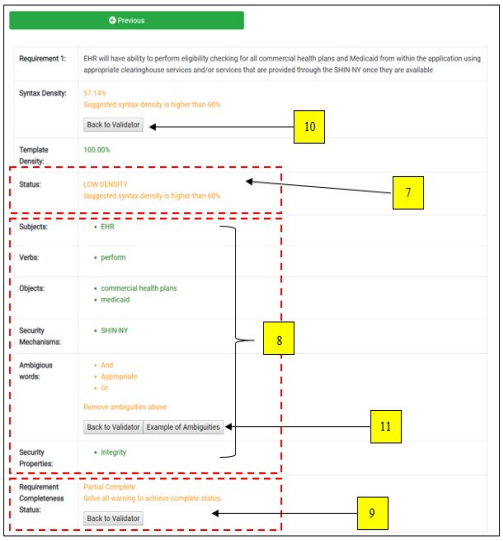


Figure 8: Checking the security requirements and key-structure components

Figure 9: Validation of completeness prioritization

5. TOOL VALIDATION

We have conducted a tool validation by making a comparison between our tool and manual studies. The main aim of this test was to evaluate the capabilities of our pattern libraries in generating complete security requirements. This completeness test involves making comparison between the number of complete security requirement and the key-textual structure components, generated by manual approach and using the SRC and SecLib pattern libraries embedded in our tool. The objective of this test is to compare the number of complete security requirement and key-textual structure components generated by manual approach with our SRC and SecLib pattern libraries.

The manual approach data was collected from the result of our study, where the participants need to extract security requirements from the business scenario. The analysis involved in this completeness test is comparing the complete security requirements result from manual method with the pattern libraries embedded in SecureMEReq tool. Based on our tool comparisons (Refer to Background and Motivations), the comparison study between manual and template-based approach was conducted. It was found that there is no tool that supports automated completeness checking. Therefore, the comparison study between the manual and template-based approach was conducted as tools with similar functionalities provided by SecureMEReq were not found.

5.1 Study sample/subject

This survey was conducted with 68 undergraduate students who enrolled in the course of Software Validation and Verification. These students were majoring in Software Development at the *Universiti Teknikal Malaysia Melaka*. They have sufficient background and knowledge to understand about software requirements because they have already taken the Software Engineering and Software Requirement and Design subjects. The participants of this survey were volunteers and their participations were treated anonymously. These students were reliable as participants instead

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of professional software as they found that the differences are only minor to draw conclusions (Höst et. al., 2000). Table 4 summarizes the demographic details of the participants.

Table 4: The Demography Details of The Survey Participants

	Demography Information
Number of Participants	68
Level of Study	Undergraduate
Stage of Study	Second year
Academic Major	Software Development
Purpose of Study	(1) To analyze the completeness of extracting requirements from business scenario (2) To analyze the completeness of extracting the main components of sentence structure of security requirements structure (3) To compare completeness of writing security requirements and key-textual structure components by manual between our SRC and SecLib pattern libraries. (4) To identify the difficulties of eliciting security requirements manually from the business scenario.

5.2 Study material

The study material consists of a tutorial. Participants were given explanations of the key concepts used throughout the tutorial. We have also provided theoretical and example lessons on how to extract security requirements and security requirements sentence structure.

5.3 Study procedure

We have defined and followed a simple procedure to carry out the evaluation. The main task in this study was to manually extract security requirements and key textual sentence structure from business scenario. Prior to that, the participants were given a short description of the evaluation. We provided a tutorial that explains the concepts of requirement, security requirements, key textual sentence structure and security properties in detail and gave an example of the process of extracting requirements from business scenarios. We gave them 20 minutes to understand the concept and example as given in the tutorial. Then, the evaluation went through the following steps:

The participants need to write down their start time on the provided sheets. They were given an hour to complete the task.

The participants were required to write down the security requirements on the provided sheets. To reduce the complexity and time taken, the subjects only need to write down the security requirements, key textual sentence structure and security properties.

Once completed, the participants need to write down the end time and call the researcher to submit their work.

5.4 Data collection and analysis

To measure the manual effort, we calculated and averaged the time taken of the participants to finish the task. Then, we checked and compared each of the security requirements written by the participants with our security requirements extracted from the tool to measure the completeness of their answers. For this, we checked the completeness of the participants' responses with our pattern libraries. We gave the relevant point for each response following the completeness measurements as described in Table 5.

Table 5: Completeness Measurement

No.	**Module**	**Pattern Library**	**Participant Response**
1.	Extract Requirements	4	4 matches = complete 3, 2, 1 or 0 matches = incomplete
2.	Identify Subject/Verb/Object	3	3 matches = complete 2,1 or 0 match = incomplete
3.	Identify Security Mechanism	1	1 match = complete 0 match = incomplete
4.	Identify Ambiguous Word	1	1 match = complete 0 match = incomplete
5.	Identify Security Properties	1	1 match = complete 0 match = incomplete

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6. THREATS OF VALIDITY

The evaluation of the proposed tool is positioned within the positivist realm, in which the focus is to maintain objectivity and unbiased results. As such, there are threats of validity related to the testing and evaluation methods that need to be addressed and acknowledged. There are two types of threats to validity: (a) internal threats and (b) external threats.

Internal validity threats are experimental procedures, treatments, or experiences of the participants that threaten the researcher's ability to draw correct inferences from the data about the population in an experiment. Nevertheless, potential threats to external validity also must be identified and designs created to minimize these threats. External validity threats arise when experimenters draw incorrect inferences from the sample data to other persons, other settings, and past or future situations [29].

In this section, we discuss the types of validity taken care in each of testing and evaluation approaches: (1) survey, (2) observation and (3) interview. To address the sampling bias, we have selected undergraduate students majoring in Software Development. In general, they have the same level of knowledge and skill in software requirements. To address the historical effect, we made sure that all participants conducted the evaluations at the same time and the same place. We provided a tutorial and tool demonstration prior to the evaluations to ensure that the participants were properly trained and have sufficient theoretical knowledge. They were also not aware of the main objective of the evaluation. With respect to maturity effect, we informed the participants that their response will be treated anonymously and they were not evaluated on their performance. For this, we provided a written informed consent forms to each of the participants. They need to read and sign to express their consent to participate in the study, before they begin the evaluations.

There were two forms of threat identified in both observation and interview approach, which are the descriptive and interpretive validity. Descriptive validity refers to the factual accuracy of the descriptive information gathered from a particular phenomenon, situation or group. To address the descriptive validity, we used multiple observers to collect and interpret the data. The observers were postgraduate students who have sufficient experience conducting observation research. The participants were requested to do the manual elicitation and to explore the tool. Here, we assigned two observers in the laboratory to observe the participants' communication and behavior during the session. The results from the observation were discussed and agreed by each observer. Similarly, for the interview, we asked the permission to tape-recording the interview session. This helps us to transcribe and analyze the result from the interview.

The second type of validity is the interpretive validity. It refers to the degree that the researcher accurately understands the participants' viewpoint and thought. In order to avoid compromising interpretive validity, we used open-ended questions to obtain the participants' feedback upon completion of the task.

7. RESULTS AND DISCUSSION

Responding to RQ in Section 3, we have conducted an evaluation. Table 6 shows the results of comparison analysis between the manual approach and our tool. The objective of this test was to compare the number of complete security requirements and key-textual structure extracted using the manual approach and our template-based approach pattern libraries embedded in our tool.

The result in Table 6 shows that the mean completeness from the manual approach was 9.8%, in which 90.2% was the incomplete answers. Meanwhile, the result from SecureMEReq shows 100% completeness for the extraction of security requirements and key-textual structure with 0% for incomplete answer.

Most of them (72%) were having difficulty to manually extract the security requirements and key-textual structure. Only approximately a quarter (28%) of them could easily extract correct requirements from the business scenario, even though they understood the business scenario given. From the result, most of them failed to identify the security mechanism correctly and none of them failed to identify the ambiguous words and security properties correctly. These are due to the fact that they were not familiar with security requirements and lacked of experience handling security requirements. They felt difficult and needed more time to manually identify the security information from the business scenario. As overall, they felt that manual elicitation is time consuming and they needed experience to do the task. From this result, we have also found that the template-based approach performs better than the manual approach. Furthermore, the tool can generate complete

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

security requirements in just over seconds in comparison to the manual approach, which takes an average of 16.4 minutes to complete the tasks.

Table 6: Results from Comparison of Manual and SecureMEReq

Task	No. of Complete Answers (%)		No. of Incomplete Answers (%)	
	Manual	SecureMEReq	Manual	SecureMEReq
Extract Requirements	28	100	72	0
Identify Subject/ Verb/ Object	12	100	88	0
Identify Security Mechanism	9	100	91	0
Identify Ambiguous Word	0	100	100	0
Identify Security Properties	0	100	100	0
Mean	**9.8**	**100**	**90.2**	**0**

8. CONCLUSION AND FUTURE WORKS

In summary, we have presented our prototype tool, called SecureMEReq that provides the (1) extraction of template-based components from client-stakeholders; (2) analysis of template-based density from SRCLib; (3) analysis of requirements syntax from SecLib; and (4) analysis of completeness prioritization. Our evaluation results show that our prototype tool was able to produce the complete security requirements in comparison to manual task and this answered to our research question that aimed to evaluate the usefulness of SecureMeReq in writing complete security requirements. Based on evaluation conducted, the tool was able to reduce the manual effort and the participants agreed that this tool can facilitate the writing of the complete security requirements. This feature helps to accelerate the writing of security requirements process and reduce the development cost. For future research, we will extend the evaluation of our tool by evaluating the efficacy of our approach in terms of completeness. We will conduct completeness testing to evaluate the completeness of eliciting security requirements by comparing manual elicitation with our prototype tool. This is to determine the ability of our SecureMEReq tool to produce complete security requirements. We strongly believe that our template-based approach is able to enhance the clarity of requirements that leads to completeness of writing security requirements and contribute to the success of secure software development.

ACKNOWLEDGEMENTS

We would like to express our gratitude to the university, UTeM and MoHE for the research funding: PASCA-COVID19/2020/FTMK-CACT/C00001

REFERENCES:

[1] K. Schneider, E. Knauss, S. Houmb, S. Islam, and J. Jürjens, "Enhancing Security Requirements Engineering by Organizational Learning," in *Requirements Engineering*, vol. 17, no. 1, 2012, pp. 35–56.

[2] D. G. Firesmith, "Engineering Safety And Security Related Requirements For Software Intensive Systems," in *29th International Conference on Software Engineering (ICSE 2007)*, 2007, p. 169.

[3] E. Kamsties and B. Paech, "Taming Ambiguity in Natural Language Requirements," in *the Thirteenth International Conference on System and Software Engineering and their Applications*, 2000.

[4] M. Bano, "Addressing the Challenges of Requirements Ambiguity: A Review of Empirical Literature," in *5th International Workshop on Empirical Requirements Engineering (EmpiRE 2015)*, 2016, pp. 21–24.

[5] M. Kamalrudin, N. Mustafa, and S. Sidek, "A Preliminary Study: Challenges In Capturing Security Requirements And Consistency Checking By Requirement Engineers," *J. Telecommun. Electron. Comput. Eng.*, vol. 10, no. (1-7), pp. 5–9, 2017.

[6] J. Slankas, M. Riaz, J. King, and L. Williams, "Discovering Security Requirements from Natural Language Project Artifacts," in *36th International Conference on Software Engineering*, 2014,

pp. 1–12.

[7] D. G. Firesmith, "Analyzing and Specifying Reusable Security Requirements," in *IEEE 11th International Conference on Requirements Engineering, RHAS 2003*, 2003, pp. 507–514.

[8] M.-L. Sánchez-Gordón, R. Colomo-Palacios, A. Sánchez, A. De Amescua Seco, and X. Larrucea, "Towards the Integration of Security Practices in the Software Implementation Process of ISO/IEC 29110: A Mapping," in *European Conference on Software Process Improvement (EuroSPI 2017)*, 2017, pp. 3–14.

[9] G. Kotonya and I. Sommerville, *Requirements Engineering : Processes and Techniques*. J. Wiley, 1998.

[10] R. Sharma and K.K. Biswas, "Resolving Inconsistency and Incompleteness Issues in Software Requirements," in *Springer Managing Requirements Knowledge*, 2013, pp. 315–332.

[11] D. Zowghi and C. Coulin, "Requirements Elicitation: A Survey of Techniques," in *Engineering and Managing Software Requirements*, Berlin, Heidelberg: Springer, Berlin, Heidelberg, 2005, pp. 19–46.

[12] R. Agarwal and M. R.Tanniru, "Knowledge acquisition using structured interviewing: an empirical investigation," *J. Manag. Inf. Syst.*, vol. 7, no. 1, pp. 123–140, 2015.

[13] W. Foddy, *Constructing questions for interviews and questionnaires*. Cambridge University Press, 1993.

[14] A. M. Hickey and A. M. Davis, "Elicitation Technique Selection : How Do Experts Do It ?," in *Proceedings of the 11th IEEE International Requirements Engineering Conference*, 2003.

[15] D. Wixon and J. Ramey, *Field Methods Casebook for Software Design (1st Edition)*. Wiley, 1996.

[16] M. Antonio da Silva and M. Danziger, "The Importance Of Security Requirements Elicitation And How To Do It," in *PMI® Global Congress 2015*, 2015, pp. 1–12.

[17] T. R. Farkhani and M. R. Razzazi, "Examination and Classification of Security Requirements of Software Systems," in *IEEE The 2nd International Conference on Information & Communication Technologies*, 2006, vol. 2, pp. 2778–2783.

[18] M. Zhivich and R. K. Cunningham, "The Real Cost of Software Errors," *IEEE Secur. Priv.*, vol. 2, no. 2, pp. 87–90, 2009.

[19] P. Salini and S. Kanmani, "Survey and Analysis on Security Requirements Engineering," *Comput. Electr. Eng.*, vol. 38, pp. 1785–1797, 2012.

[20] D. G. Firesmith, "Engineering Security Requirements," *J. Object Technol.*, vol. 2, no. 1, pp. 53–68, 2003.

[21] M. Riaz and L. Williams, "Security Requirements Patterns: Understanding The Science Behind The Art Of Pattern Writing," in *IEEE 2nd International Workshop on Requirements Patterns (RePa 2012)*, 2012, pp. 29–34.

[22] T. Sven, "The Trouble With Security Requirements," in *IEEE 25th International Requirements Engineering Conference (RE2017)*, 2017, pp. 122–133.

[23] D. Hatebur, M. Heisel, and H. Schmidt, "Security Engineering Using Problem Frames," in *Emerging Trends in Information and Communication Security*, 2006, pp. 238–253.

[24] D. Mellado, E. Fernández-Medina, and M. Piattini, "A Common Criteria Based Security Requirements Engineering Process For The Development Of Secure Information Systems," *Comput. Stand. Interfaces*, vol. 29, no. 2, pp. 244–253, 2007.

[25] M. Kamalrudin, J. Grundy, and J. Hosking, "Managing Consistency Between Textual Requirements, Abstract Interactions And Essential Use Cases," in *IEEE 34th International Computer Software and Applications Conference*, 2010, pp. 327–336.

[26] M. Kamalrudin, J. Grundy, and J. Hosking, "Automated Support for Consistency Management and Validation of Requirements," The University of Auckland, 2011.

[27] Ian Sommerville, *Software Engineering (10th Edition)*. Pearson, 2015.

[28] M. Höst, B. Regnell, and C. Wohlin, "Using Students as Subjects—A Comparative Study of Students and Professionals in Lead-Time Impact Assessment," in *Empirical Software Engineering*, vol. 5, 2000, pp. 201–214.

[29] J. W. Creswell, *Research Design Qualitative, Quantitative and Mixed Methods Approaches*. 2013.